基于方形领域的网格密度聚类算法 *

兰 红[+]，朱合隆

(江西理工大学 信息工程学院, 江西 赣州 341400)

摘 要：针对大数据聚类低效的问题，提出一种方形邻域快速网格密度聚类算法 SGBSCAN (square-neighborhood and Grid-based DBSCAN)。首先给出方形邻域密度聚类定义，利用方形邻域代替圆形邻域，降低时间复杂度；其次提出方形邻域密度聚类的 Grid 概念，快速确定高密度区域内核心点与数据点之间的密度关系；最后提出 Grid 密度簇，利用网格之间的关系加快密度簇的形成。算法应用于 16 个数据集，分别与已有文献算法进行对比，结果表明所提算法在聚类效率方面有显著提升，数据量越大算法效率提升越明显，且所提算法适用于多维数据的聚类。

关键词：聚类分析；密度聚类；方形邻域；Grid 网格；网格簇

中图分类号：TP274 doi: 10.19734/j.issn.1001-3695.2018.12.0883

Grid density clustering algorithm based on square neighborhood

Lan Hong[+], Zhu Helong

(*School of Information Engineering, Jiangxi University of Science & Technology*, *Ganzhou Jiangxi* 341400, *China*)

Abstract: To solve the problem of low efficiency of large data clustering, this paper proposes a fast grid density clustering algorithm SGBSCAN(Square-neighborhood and Grid-based DBSCAN) . Firstly, this paper gave the definition of square neighborhood density clustering , and used the square neighborhood instead of the circular neighborhood to reduce the time complexity. Secondly, this paper proposed the concept of grid of square neighborhood density clustering , to determine the density relationship between core points and data points in high density region quickly. Finally, this paper proposed the Grid density cluster, the method used the relationship between the grid to accelerate the formation of density clusters. The algorithm made 16 data sets and compared with the existing literature algorithms. The results shows that the algorithm has a significant improvement in clustering efficiency. The larger the data volume, the more obvious the efficiency of the algorithm, and the algorithm is suitable for multidimensional data clustering.

Key words: clustering analysis; density-based; square neighborhood; Grid; grid-based cluster

0 引言

聚类作为机器学习中无监督学习的代表，它能够在大数据下发现数据中的潜在联系，发现数据中的有价值的信息，被广泛用于生物医学邻域[1]/商业数据分析邻域[2]/位置服务[3]/图像处理等邻域[4]，使得其成为多方的研究重点。

聚类已经被研究很多年，衍生出多种聚类算法，大致可分为基于划分的算法（如 K-means[5]）、基于层次的算法（如 Chameleon[6]）、基于密度的算法（如 DBSCAN[7]和 OPTICS[8]）、基于网格的算法（如 STING[9]和 GG[10]）、基于模型的算法（如 GMM 算法[11]）。其中基于密度的算法由于其具有可以识别任意形状的簇且可以识别噪声的优点，受到如位置数据挖掘领域[12]的青睐。 其中由 Ester 等人[7]提出的 DBSCAN 为密度聚类的代表性算法，虽然 DBSCAN 有密度算法的优点，但是其 $O(n^2)$ 时间复杂度使得该算法很难应用于大规模数据的聚类。Ankerst 等人[8]提出的 OPTICS 算法使用广泛参数代替 DBSCAN 的绝对参数解决了 DBSCAN 中参数敏感性问题，但是该算法的算法框架流程与 DBSCAN 相同，都得对数据集进行重复遍历，时间复杂度依然是 $O(n^2)$。由 Wang 等人[9]提出的 STING 算法基于网格查询，效率提高明显，但是聚类质量不足。后续由 Zhao 等人[10]提出的基于网格的 GG 算法依旧存在着算法聚类精度不高的问题。

由于彦伟等人[12]提出的面向位置大数据的快速密度聚类算法有效的提高了密度聚类的效率，算法采用圆形邻域，结合 *Cell* 网格、*Cell* 距离分析和网格簇获得了显著的加速比。但是该算法使用圆形邻域获得的加速比在效率上还有待进一步提升，圆形邻域结合 *Cell* 等概念针对二维数据友好，但是不易于拓展到高维数据。现实中，高维数据聚类分析在市场分析、信息安全、金融、娱乐、反恐等方面都有很广泛的应用，高维数据的聚类分析已成为聚类分析的一个重要研究方向。

本文针对文献[12]存在的不足，提出一种改进的密度聚类算法 SGBSCASN。首先使用方形邻域替代传统的圆形邻域去除距离计算，其次采用 *Grid* 距离分析减少了距离对比次数，最后基于 *Grid* 的密度簇概念提出的网格间的关系减少数据的遍历提高了聚类效率。改进算法在七个数据集下进行了实验效果评估，在 Geolife[19~21]项目的大数据集下进行了数据规模的性能测试，在 HTRU2[22]多维大数据集下进行了数据维度的性能测试，这几项测试从算法的聚类效果和算法性能方面与 DBSCAN、K-Means 和 CBSCAN 进行了对比分析。

收稿日期：2018-12-18；**修回日期：**2019-02-18 **基金项目：**国家自然科学基金资助项目（61762046）；江西省自然科学基金资助项目（20161BAB212048）

作者简介：兰红（1969-），女（通信作者），辽宁鞍山人，教授，硕导，博士，主要研究方向为数据挖掘、图像处理（40376334@qq.com）；朱合隆（1994-），男，江西赣州人，硕士研究生，主要研究方向为数据挖掘、聚类.

1　相关定义

1.1　方形邻域密度聚类

1）基本思想

密度聚类是根据密度分布将高密度区域的数据点划分为同一类的聚类方法，该方法能够聚类出任意形状的数据集，且具有识别数据集中噪声点的能力。传统的密度聚类算法采用圆形邻域，采用欧氏距离判断数据点是否在领域内，如式（1）所示，通过欧氏距离计算生成聚类簇。

$$\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}<Eps \tag{1}$$

其中：x_i,y_i 分别表示数据点对应维度的值；Eps 为圆形领域的半径。

欧氏距离中存在多次乘运算及开方运算，时间复杂度为 $O(3n)$。

本文利用方形代替圆形，判断数据点是否在领域内采用式（2）。

$$\bigcap_{i=1}^{n}(|x_i-y_i|<\frac{Ssl}{2}) \tag{2}$$

无须进行乘运算及开方运算，时间复杂度为 $O(2n)$，提高聚类效率。

2）相关术语定义

设数据集为 D，数据点 $p\in D$，$q\in D$，首先定义 ***Ssl*** 和 $MinPts$。

***Ssl*（*Square Side Length*）**：以数据点为中心的超立方体邻域的边长。

***MinPts*（*Min Points*）**：最小邻居点的数目值。数据点 p 要成为核心点的必要条件即其邻域内至少包含 $MinPts$ 个数据点，这些点称为 p 点的邻居点。

用 $N_{Ssl}^{D}(p)$ 表示点 p 在其 Ssl 邻域内邻居点的集合，$|N_{Ssl}^{D}(p)|$ 表示该集合中数据点的数量。给出方形邻域的核心点、边界点等相关术语定义。

核心点：如果数据点 p 满足式（3），则其为核心点。

$$|N_{Ssl}^{D}(p)|\geq MinPts \tag{3}$$

边界点：如果数据点 p 满足式（4）的条件，则其为边界点。

$$|N_{Ssl}^{D}(p)|<MinPts \;\&\; 核心点\in N_{Ssl}^{D}(p) \tag{4}$$

噪声点：如果数据点 p 满足式（5），则其为噪声点。

$$|N_{Ssl}^{D}(p)|<MinPts \;\&\; 核心点\notin N_{Ssl}^{D}(p) \tag{5}$$

直接密度可达：设 q 为核心点，且 $p\in N_{Ssl}^{D}(q)$，则说 p 从 q 出发是直接密度可达的。可以得出若 $p\in N_{Ssl}^{D}(q)$，则 $q\in N_{Ssl}^{D}(p)$。

密度可达：给定数据点集合 $\{p_1,p_2,\cdots,p_n\}\subseteq D$，若任意 p_{i+1} 从 $p_i(i=1,2,\cdots,n-1)$ 出发是直接密度可达的，则称 p_1 密度可达 p_n。

密度相连：若存在数据点 $o\in D$，且 p 密度可达 o、q 密度可达 o，那么说 p 和 q 是密度相连的。

3）方形邻域密度簇

由以上术语的定义，给出方形邻域密度簇的定义。

定义 1　方形邻域密度簇。给定数据集 D，将方形邻域内密度相连的点的非空集合定义为密度簇。

1.2　基于 *Grid* 划分的距离分析

针对方形邻域密度聚类，提出基于网格划分的距离分析方法，称为 *Grid* 网格划分，给出基于网格层数参数 *Layer*的*Grid* 定义，提出基于 *Grid* 的距离关系分析方法及最优 *Layer* 选择问题。

1）网格　***Grid*** 定义

Layer：基于 *Grid* 的距离分析需要在方形邻域密度聚类的基础上增加网格层数参数 $Layer(Layer\geq 3)$，数据点的方形邻域称为 *Square*，*Layer* 定义 *Square* 所包含的网格层数。

定义 2　***Grid***。给定数据集 D、参数 Ssl，将数据集 D 划分到网格中，网格边长 GL 的计算公式为

$$GL=\frac{Ssl}{2Layer-1} \tag{6}$$

每个划分的网格称为 *Grid*，表示为 G_I，其中下标 I 的值通过式（7）计算获得；X 为数据点的坐标向量。

$$\mathrm{I}=\left\lfloor\frac{(2\mathrm{Layer}-1)\cdot\mathrm{X}}{\mathrm{Ssl}}\right\rfloor \tag{7}$$

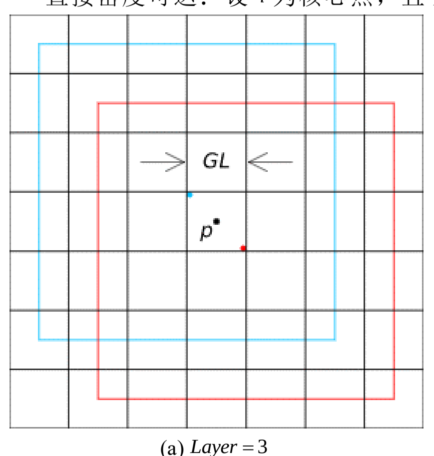
由 *Grid* 的定义可知，对于任何数据点，都可以使用式（7）将其映射到相应的 *Grid*，所以根据数据点坐标，即可快速定位到相应的网格。

2）　网格 ***Grid*** 与核心点的关系

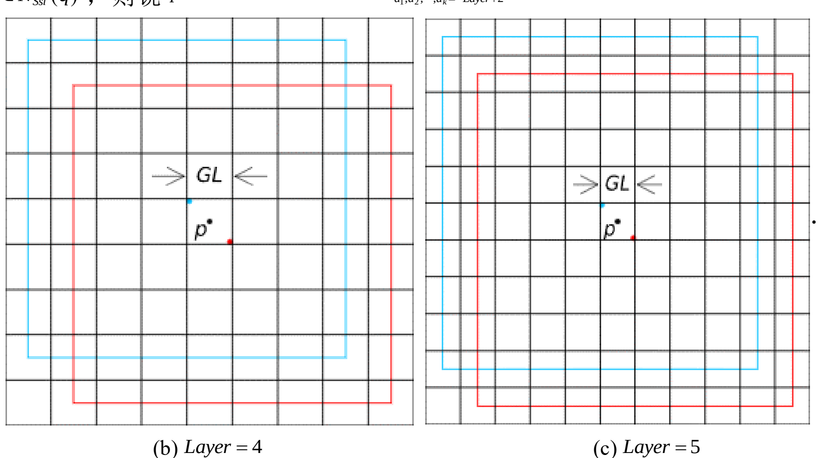
图 1 所示为 *Layer* 与网格的关系。下面分析网格 ***Grid*** 与核心点之间的关系。

引理 1　给定网格 G_I 中的一个数据点 p，设 p 为 k 维数据点，则 p 点的坐标为 $X=[x_1,x_2,\cdots,x_k]$，定义 $|G|$ 表示网格 G 中的数据点数量，若 G_I 满足式（8），则点 p 一定是核心点。（证明见附录）

$$\sum_{d_1,d_2,\cdots,d_k=-Layer+2}^{Layer-2,Layer-2,\cdots,Layer-2}|G_{(i_1-d_1,i_2-d_2,\cdots,i_k-d_k)}|\geq MinPts \tag{8}$$



(a) $Layer=3$



(b) $Layer=4$　　(c) $Layer=5$

图 1　基于 Grid 网格的距离分析

Fig. 1　Distance analysis based on Grid

推广引理 1 可得出推论 1。

推论 1 给定一个网格 G_l，如果其满足式（8），则 $\forall p(p\in G_l)$ 都是核心点。

更可以推出下面结论：如果 G_l 满足式（9），

$$|G_l|\ge MinPts \tag{9}$$

则

$$\forall G(G\in\{G_{(i_1-d_1,i_2-d_2,\cdots,i_k-d_k)}\mid -Layer+2\le d_1,d_2,\cdots,d_k\le Layer-2\})$$

G 内所有的数据点都是核心点。

引理 2 给定数据点 $p\in G_l$，判断 p 是否为核心点，则至多需要判断 $(2Layer)^k-(2Layer-3)^k$ 个网格中为 p 的邻居点的数据点个数。（证明见附录）

引理 2 说明了验证 p 是否为核心点的最大搜索范围。以引理 2 可以推出以下推论 2，根据推论 2，可以快速地对网格是否存在核心点进行排除。

推论 2 给定网格 G_l，如其满足式（10），则 G_l 内不存在核心点。

$$\sum_{d_1,d_2,\cdots,d_k=-Layer}^{Layer,Layer,\cdots,Layer}|G_{(i_1-d_1,i_2-d_2,\cdots,i_k-d_k)}|<MinPts \tag{10}$$

3） ***Grid* 中选择最优 *Layer***

算法中存在 *Layer* 的选择问题，选择合适的 *Layer* 值将带来算法性能的提升。

下面分析 *Grid* 中选择最优 *Layer* 的问题。

设 A 为事件，事件内容为利用引理 1 判断网格内所有的点是否为核心点，则事件 A 发生的概率为

$$P(A)=\frac{(2Layer-3)^k}{(2Layer)^k} \tag{11}$$

可以看出，当 *Layer* 增大，$P(A)$ 也增大；但另一方面，划分的网格数量为

$$N_t=\frac{S(2Layer-1)^k}{Ssl^k} \tag{12}$$

其中：S 表示所有数据点邻域占据的空间；k 表示维度。随着 *Layer* 增大，N_t 也在增大。

当 $(k\ge 3)$ 时，N_t 增长速度远大于 $P(A)$ 的增长速度，划分的网格数量多影响网格的索引效率。因而 *Layer* 建议选择为 3，在高密度下可选择 4 以加快数据聚类。实验 4.3.3 节测试了 *Layer* 的取值对算法效率的影响。

2 基于 *Grid* 的密度聚类算法

将 *Grid* 应用于方形邻域，提出基于 *Grid* 的密度簇概念。

2.1 基于 *Grid* 的密度簇

全核心网格（ACG）：给定网格 G_l 和密度簇 C，若 $\forall p\in C(p\in G_l)$ 且 $\forall p$ 都是核心点，则称网格 G_l 为全核心网格（all core grid，ACG）。

由定义可知，对于 G_l，其能够成为全核心网格的充要条件为 G_l 满足式（8）。

半核心网格（HCG）：给定网格 G_l 和密度簇 C，若 $\forall p\in C(p\in G_l)$ 且 $\exists p$ 为核心点，则称网格 G_l 为半核心网格（half core grid，HCG）。

由定义可知，对于 G_l，其能够成为半核心网格的充要条件为：a) G_l 为全核心网格或者 b) G_l 满足式（10）且根据引理 2 可确定 $\exists p$ 为核心点。可以看出，全核心网格 $\subset$ 半核心网格。

非核心网格（NCG）：给定网格 G_l 和密度簇 C，若 $\forall p\in C(p\in G_l)$，则称网格 G_l 为非核心网格（non core grid，NCG）。

由定义可知，对于 G_l，其能够成为非核心网格核的充要条件为：a) G_l 为半核心网格或者 b) G_l 的中心点的 *Ssl* 邻域内存在半核心网格，即如果有 G_{l_2}（G_{l_2}为半核心网格）在 G_l 的中心点的 *Ssl* 邻域内，则 G_l 为非核心网格。G_{l_2} 为半核心网格，则一定 $\exists p(p\in G_{l_2})$ 为核心点，则 G_l 内所有的点都是密度可达 p，即 $\forall p\in C(p\in G_l)$。

边界网格（BG）：给定网格 G_l 和密度簇 C，若 $\exists p\in C(p\in G_l)$，则称网格 G_l 为边界网格（boundary grid，BG）。

SGBSCASN 算法即在一定的密度簇上聚合全核心网格、半核心网格、非核心网格以及边界网格内的所有的边界点。算法应用推论 1，只需统计单个网格数据点数量，便能大范围的确定全核心网格；应用引理 1，只需统计 $(2Layer-3)^k$ 个网格中的数据点个数，无须进行距离计算，即可快速确定数据中的高密度区域以及全核心网格；应用引理 2，只需少量的范围判断以及统计网格数量，便可快速地推断出网格中数据点是否为核心点；推论 2 的应用可以快速地确定网格内是否存在核心点，以减少计算；由定义 5 可以快速确定边界网格区域，快速对边界点进行聚类。算法做到了在保证算法精确度的情况下完全不需要距离计算，即可快速地确定密度簇。

2.2 SGBSCAN 算法描述

SGBSCAN 算法的总体流程如图 2 所示。首先依据输入的参数将数据集中的数据点划分到各个网格中，并且得到各个网格的索引。其次，遍历所有的数据点，通过数据点获得数据点所在网格，判断数据点和网格是否未访问。如果未访问，则判断其是否为 *ACG* 或者 *HCG*。若是则拓展密度簇，若不是则认为网格属于 *NCG* 或者 *BG*，其内的数据点属于边界点或噪声，然后自增 *Cluster*。循环遍历至结束，返回聚类结果。

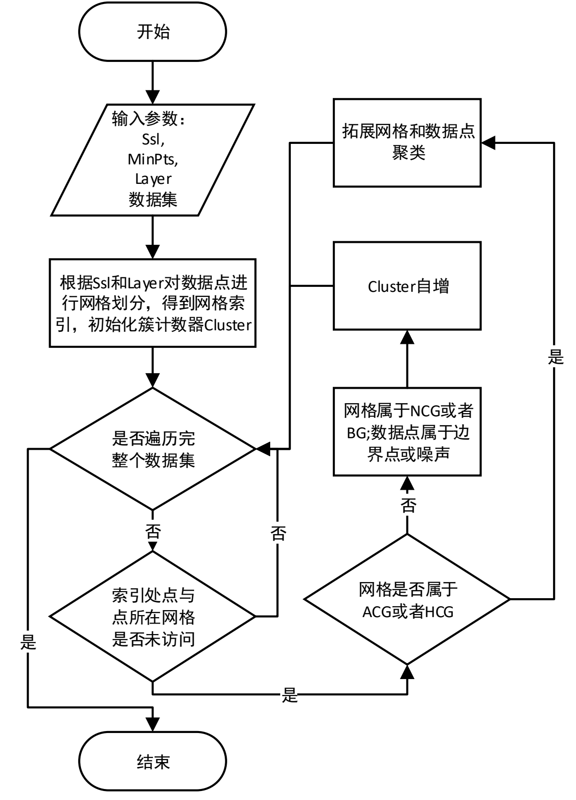


图 2 算法总体流程

Fig. 2 Algorithm overall flow

2.3 SGBSCAN 算法实现

算法 1 描述了算法实现的总框架。

算法 1　SGBSCAN 算法框架

```
Algorithm1 SGBSCAN
Parameter:Ssl,MinPts and Layer
Input:Dataset D.
Output:dataset-with-cluster-id DWCI.
Algorithm:
1.Cluster ← 1;
2.Grids ← divideDataIntoGrid(D,Ssl,Layer,MinPts);
3.for each p in D do
4.    G ← gridOfPoint(p);
5.    if p.isUnvisited && G.isUnvisited than
6.        if Σ|G| ⩾ MinPts // G and it's neighbor grid are ACG
7.            G.cluster ← neighborGrid(Layer,G).cluster ← Cluster;
8.            G.visited ← neighborGrid(Layer,G).visited ← true;
9.            GridsToVisit.add(boundaryGrid(Layer,G));
10.           EXPANDGRIDS(Ssl,Layer,MinPts,Cluster,GridsToVisit);
11.           Cluster++;
12.       else if (Σ|G| + Σ|neighborGrid(Layer,G)|) ⩾ MinPts than
// G is ACG
13.           G.cluster ← Cluster;G.visited ← true;
14.           GridsToVisit.add(neighborGrid(Layer,G));
15.           EXPANDGRIDS(Ssl,Layer,MinPts,Cluster,GridsToVisit);
16.           Cluster++;
17.       else if (Σ|G| + Σ|neighborGrid(Layer,G)| +
18.           Σ|inSslOfPoint(Ssl,p,neighborGrid(Layer,G))|) than
// G is HCG
19.           p.cluster ← Cluster;p.visited ← true;
20.           GridsToVisit.add(neighborGrid(Layer,G));
21.           EXPANDGRIDS(Ssl,Layer,MinPts,Cluster,GridsToVisit);
22.           PointsToVisit.add(boundaryGrid(Layer,G));
23.           EXPANDPOINTS(Ssl,Layer,MinPts,Cluster,PointsToVisit);
24.           Cluster++;
25.       else
26.           p.cluster ← Noise; // p maybe a boundary point or noise
27.       end if
28.    end if
29.end for
30.return DWCI;
```

首先初始化 *Cluster* 计数器（1 行），依据输入的参数使用 *divideDataIntoGrid*() 函数对数据进行网格划分得到 *Grids*（2 行）。

然后依次遍历所有数据集中的数据点 p，通过 *gridOfPoint*() 函数获取 p 所在的网格 G。先判断 p 和 G 是否未访问，若未访问，则判断其是否为 *ACG* 或者 *HCG*。若是则拓展密度簇（6-24 行）；若不是，则置为噪声（25~26 行）。如果 p 不是噪声，在后续拓展中会将其分类为边界点。

其中：

GridsToVisit 为用于存储待拓展的网格的队列；

PointsToVisit 为用于存储待拓展的数据点的队列；

neighborGrid(*Layer*,*G*) 表示依 *Layer* 参数界定的 G 的所有邻居网格；

boundaryGrid(*Layer*,*G*) 表示依 *Layer* 参数界定的 G 的所有边界网格；

inSslOfPoint(*Ssl*, *p*, *neighborGrid*(*Layer*,*G*)) 表示 G 的所有边界网格内的所有在 G 邻域的数据点的集合。

算法 1 中的 EXPANDGRIDS 函数用于实现拓展网格密度簇，其函数定义见算法 2。

算法 2　**EXPANDGRIDS 函数**

```
FUNCTION
EXPANDGRIDS(Ssl,Layer,MinPts,Cluster,GridsToVisit)
1.while GridsToVisit.isNotEmoty do
2.    G ← GridsToVisit.poll();
3.    G.cluster ← Cluster;G.visited ← true;
4.    if Σ|G| ⩾ MinPts than
5.        neighborGrid(Layer,G).cluster ← Cluster;
6.        neighborGrid(Layer,G).visited ← true;
7.        GridsToVisit.add(boundaryGrid(Layer,G));
8.    else if (Σ|G| + Σ|neighborGrid(Layer,G)|) ⩾ MinPts than
9.        GridsToVisit.add(neighborGrid(Layer,G));
10.   else if (Σ|G| + Σ|neighborGrid(Layer,G)| +
11.       Σ|inSslOfPoint(Ssl,p,neighborGrid(Layer,G))|) than
12.       p.cluster ← Cluster;p.visited ← true;
13.       GridsToVisit.add(neighborGrid(Layer,G));
14.       PointsToVisit.add(boundaryGrid(Layer,G));
15.       Cluster++;
16.   end if
17.end while
```

循环访问 *GridsToVisit* 直到其为空。首先获取并移除 *GridsToVisit* 队列的头，赋值为 G（2 行），并设 *G.cluster* 为 *Cluster* 且已访问（3 行），再判断其是否为 *ACG*。如果是，则执行将其邻居网格压入队列（4-9 行）；如果不是，则其可能为 *HCG*, *NCG*, *BG*，将其邻居网格压入队列，且将边界网格内的数据点压入 *PointsToVisit* 队列（10~15 行）。

算法 1 中的 EXPANDPOINTS 函数主要对非噪声点的边界点进行判定。其函数定义见算法 3。

算法 3　EXPANDPOINTS 函数

```
FUNCTION
EXPANDPOINTS(Ssl,Layer,MinPts,Cluster,PointsToVisit)
1.while PointsToVisit.isNotEmoty do
2.    p ← PointsToVisit.poll();
// If p is a boundary point, it will be classified
3.    p.cluster ← Cluster;p.visited ← true;
4.    G ← gridOfPoint(p);
5.    if (Σ|G| + Σ|neighborGrid(Layer,G)| +
6.        Σ|inSslOfPoint(Ssl,p,neighborGrid(Layer,G))|) than
7.        PointsToVisit.add(boundaryGrid(Layer,G));
```

```
8.    end if
9.end while
```

2.4 复杂度分析

设给定数据集的数据点数量为 n，对数据点进行网格划分后的网格数量为 m，若 γ 为边界点和噪声点占数据集的比例，则数据集中边界点和噪声点的数量为 $\gamma \cdot n$；若 α,β 分别为全核心网格和半核心网格占总网格数的比例，则数据集中全核心网格和半核心网格的数量分别为 $\alpha \cdot m,\beta \cdot m$。

设 $B=(2Layer)^k, N=(2Layer-3)^k$，则本文 SGBSCAN 算法的时间复杂度为

$$O(n)+O(\alpha \cdot m \cdot N)+O(\beta \cdot m \cdot B)+O(\gamma \cdot n \cdot B) \tag{13}$$

其中：

$O(n)$ 为网格划分的时间复杂度；$O(\alpha \cdot m \cdot N)$ 为遍历全核心网格的时间复杂度，识别一个网格是否为全核心网格,只需要统计其 N 个邻居网格；$O(\beta \cdot m \cdot B)$ 为遍历半核心网格的时间复杂度，判别一个网格是否为半核心，则需要统计其 N 个邻居网格加判别 $B-N$ 个边界网格内在数据点 Ssl 邻域内的数据点个数，总的来说，即需要扫描 B 个网格；$O(\gamma \cdot n \cdot B)$ 为遍历边界点和噪声点的时间复杂度，判别一个边界点和噪声点最少需要扫描 B 个网格。

3 实验及分析

对 SGBSCAN 的聚类效果和效率进行综合实验评估，并且加入 DBSCAN、CBSCAN 和 K-means 进行对比分析。实验平台为 CPU：Intel(R) Core(TM) i7-4710MQ @ 2.5 GHZ，Memory：16 GB，Operating System：Windows 10 version 1803 x64，Programming Language：Java。

3.1 实验数据和测试方法

实验数据：数据如表 1 所示，共包含 16 个数据集。前 7 个数据集（从 Flame 到 t4.8k）用于展示聚类效果。其中 Flame[13] 包含一个凸数据集和一个凹数据集的密度簇；Pathbased[14] 包含一个环形簇和两个内嵌的高斯密度簇；Compound[15]包含 6 个不同形状的复杂结构密度簇，密度簇之间关系包括相连、内嵌、包含和不同密度重叠的情况；R15[16]由 15 个大小相似的高斯密度簇组成；Aggregation[17] 包括了 7 个不同形状的非高斯分布的密度簇；D31[16]包含 31 个高斯密度簇；t4.8k[18]由 7 个不同形状、互相嵌套，并且掺杂了大量噪声的密度簇构成。

User 等 8 个数据集用于测试数据规模对性能的影响。数据来自微软亚洲研究院的 Geolife 项目[19~21]。该项目数据集包含 182 个用户在超过五年（从 2007 年 4 月到 2012 年 8 月）时间里的户外活动。该数据集的 GPS 轨迹由一系列带有时间戳的点表示，每个点包含纬度、经度和高度信息。选用 Geolife 项目数据集的纬度和经度信息。User000 表示为使用了编号为 000 的用户所有活动数据，User000-002 表示使用编号从 000~002 的 3 个用户的所有活动数据，其他数据集意思相同。

HTRU2[22] 数据集用于测试数据维度对性能的影响。HTRU2 数据集描述在宇宙测量期间收集的脉冲星候选物样本的数据集。 数据集一共有 9 个维度，包含 1 639 个正面例子；16 259 个负面例子。

3.2 实验效果评估

对表 1 的前 7 个二维数据集（从 Flame 到 t4.8k）用图 3 进行展示聚类效果。分别用本文的 SGBSCAN 算法、传统的 DBSCAN 算法、文献[12]的 CBSCAN 和 K-means 算法进行对比分析。其中 DBSCAN 和 CBSCAN 参数 Eps 的取值为表 1 中对应的 $\sqrt{\frac{Ssl^2}{\pi}}$，MinPts 取表 1 中的值；SGBSCAN 采用表 1 中的所有参数；K-means 的参数 K 采用表 1 中的 *Classes* 的值。

表 1　实验数据集

Table 1　Experimental data set

Dataset	# Instances	Dimension	Classes	Parameters
Flame	240	2	2	Ssl=2.48, Layer=3, MinPts=10
Pathbased	300	2	4	Ssl=3.37, Layer=3, MinPts=3
Compound	399	2	6	Ssl=2.48 ,Layer=3 ,MinPts=3
R15	600	2	15	Ssl=0.89 ,Layer=3, MinPts=15
Aggregation	788	2	7	Ssl=3.02, Layer=3, MinPts=10
D31	3100	2	31	Ssl=1.07, Layer=3, MinPts=15
t4.8k	8000	2	7	Ssl=17.73 ,Layer=3, MinPts=20
User018	47279	2	Unknown	Ssl=0.01064, Layer=3, MinPts=20
User000	176182	2	Unknown	Ssl=0.01064, Layer=3 ,MinPts=25
User002	248217	2	Unknown	Ssl=0.01064, Layer=3, MinPts=28
User000-002	528382	2	Unknown	Ssl=0.01064, Layer=3, MinPts=30
User000-020	5174773	2	Unknown	Ssl=0.01064, Layer=3, MinPts=50
User000-030	7649862	2	Unknown	Ssl=0.01064, Layer=3, MinPts=75
User000-040	9262868	2	Unknown	Ssl=0.01064, Layer=3, MinPts=100
User000-050	11113351	2	Unknown	Ssl=0.01064, Layer=3, MinPts=125
HTRU2	17898	9	Unknown	Ssl=1.6, Layer=3, MinPts=35

为了清晰地展示实验效果，实验效果图采用不同的颜色和不同的形状代表不同的簇，其中噪声采用灰色圆点表示。从图 3 所示的实验效果可以看出，DBSCAN、CBSCAN、SGBSCASN 由于其为精确密度聚类，所以均能很好地识别噪声，而 K-means 把噪声识别为簇的一部分，不能很好地区分噪声与正常值。K-means 由于其不能识别凹数据集，所以在数据集内有内嵌和包含关系的时候不能很好地区别簇之间的关系，而 SGBSCASN 由于其继承自 DBSCAN，拥有 DBSCAN 可以识别任意形状的簇的能力，既能识别凸集，也能识别凹集的能力，所以不论数据集内是否存在复杂的相连、内嵌、包含和不同密度重叠的关系，都能很好地进行簇的区分。其表现如图 3 (c)所示，DBSCAN、CBSCAN、SGBSCASN 均能很好地识别左下角嵌套的两个簇，而 K-means 则将两个簇从中切开；内嵌关系的表现在 t4.8k 数据集上很好地展示出来，在两个“凹”字形数据集互相嵌套，K-means 无法识别这类簇结构，采用将其切开的方式识别簇，SGBSCASN 能完美的识别该簇结构。

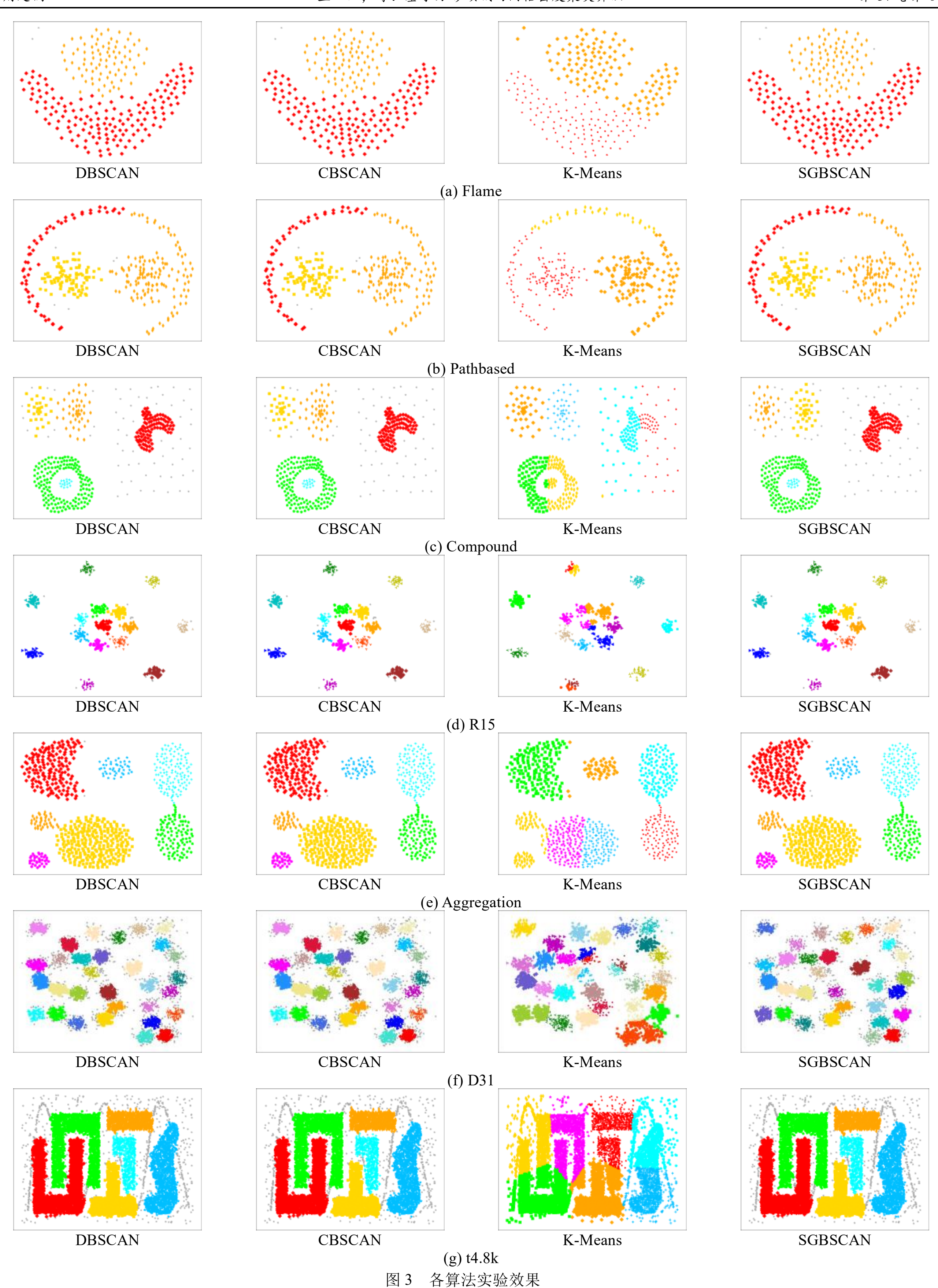


图 3　各算法实验效果

Fig. 3　Experimental results of each algorithm

3.3　性能测试

本节对数据的规模和纬度分别做测试。由于需要对比 CBSCAN，在数据规模测试时采用二维数据进行测试。在数据维度测试中，因为 CBSCAN 不支持高维数据，所以选择对比 DBSCAN。

3.3.1 数据规模性能测试

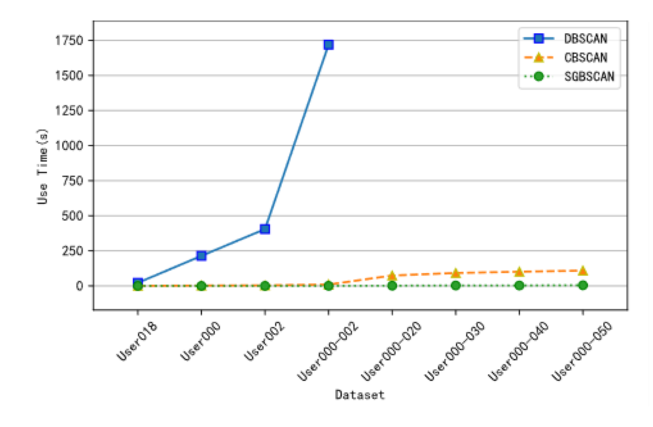
对表 1 中的 User 等 8 个数据集测试，对比 DBCAN、CBSCAN 和 SGBSCASN 算法的性能，对每个数据集采用表 1 的参数。SGBSCASN 的 *Layer* 参数为 3。各算法聚类使用时间如表 2 所示，使用时间趋势图如图 4(a)所示，内存消耗

情况如图 4(b)所示。

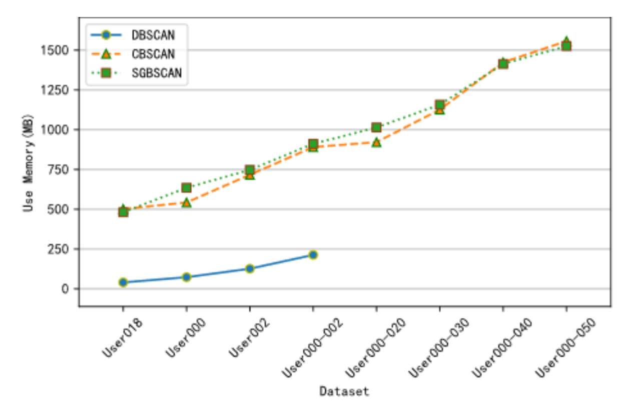
表 2　数据规模性能测试结果/S

Table 2　Data size performance test results /s

dataset	DBSCAN	CBSCAN	SGBSCASN
User018	21.91	0.39	0.023
User000	214.06	2.48	0.035
User002	405.20	3.88	0.037
User000-002	1719.57	9.71	0.173
User000-020	155976.92	73.99	1.727
User000-030	×	92.13	3.021
User000-040	×	101.04	3.167
User000-050	×	109.34	4.667



(a)Use time



(b)Use memory

图 4　性能对比

Fig. 4　Performance comparison

由表 2 中数据及图 4(a)中的趋势可知，随着数据集的数据量的增大，各种算法所用时间也随之增大。以 User000-020 数据集为例，数据量为 500 万级， DBSCAN 算法的运行时间为 155 976.92 s，耗时远大于另外两种算法；对于 User000-050 的数据集，数据量为 1 000 万级，对比 CBSCAN 和 SGBSCASN， SGBSCASN 算法能获得大约 40 倍的加速比，聚类速度更快。

分析其原因，在二维数据集下，CBSCAN 在判定排他网格中的 XG^2 网格需要检查 36 个网格，而 SGBSCASN 只需要检查 27 个网格，且 CBSCAN 在算法结构上存在冗余计算的问题。

由图 4(b)中的内存消耗趋势图可知，随着数据集的数据量的增大，各种算法内存消耗也随之增大。其中 DBSCAN 内存消耗增加的幅度变化不大，因为 DBSCAN 没有建立索引，只是循环遍历数据集；而 CBSCAN 和 SGBSCASN 由于都建立了网格索引，网格越多，建立的索引越多，内存消耗越大，且随着数据集增大内存消耗增加明显。

3.3.2 数据维度性能测试

采用表 1 中的最后一个数据集 HTRU2 测试高维度数据性能。对比 DBSCAN、K-means 和 SGBSCASN 三种算法。其中 DBSCAN、SGBSCASN 的参数取值如表 1 所示，K-Means 的参数 K 取 2。实验结果如表 3 所示，其中 Dimension 的值代表维度。

表 3　数据维度性能测试结果/s

Table 3　Data dimensional performance pest results /s

Dimension	DBSCAN	K-Means	SGBSCASN
2	8.02	17.06	0.17
3	9.71	17.53	0.41
4	13.61	18.98	1.81
5	12.78	20.78	7.95
6	14.02	23.37	54.68

从表 3 数据可以看出，与 DBSCAN 和 K-means 算法一样，SGBSCASN 算法支持高维度数据聚类。当维度小于 6 时，SGBSCASN 算法的聚类速度明显快于另外两种算法；当维度大于等于 6 时，速度较另外两种算法慢。

3.3.3 Layer 取值性能测试

本文的 SGBSCASN 算法的关键在于 *Layer* 的取值，采用 User000-020 大数据集对 *Layer* 的取值进行测试。分别设定不同的 Ssl 和 MinPts 值。实验结果如表 4 所示。

表 4　Layer 取值性能测试结果/s

Table 4　Layer value performance test result /s

Parameters \ Layer	3	4	5	6
Ssl=0.01064 MinPts=50	1.832	1.512	2.403	2.448
Ssl=0.02064 MinPts=90	1.755	1.260	2.239	2.009
Ssl=0.03064 MinPts=130	1.688	1.130	1.579	1.587

由表 4 可见，一般情况下 *Layer* 取值为 3 是最优的选择，在 *Ssl* 和 *MinPts* 值增大时， *Layer* 取值为 4 加快聚类速度，所以 *Layer* 取值与 *Ssl* 和 *MinPts* 值相关。

4　结束语

本文针对大数据聚类提出了基于密度和网格划分的快速高效的聚类算法 SGBSCAN，方形邻域和 *Grid* 的距离分析使得算法无须距离计算，提高了算法速度；基于 *Grid* 的密度簇概念给出的几种网格之间的关系和网格划分建立索引，使得算法无须遍历全部数据，提高了算法效率。在实验综合评估后，相比于传统的 DBSCAN，算法拥有更高的效率，速度提升明显，相比于文献[12]的 CBSCAN，算法能够处理高维度数据，并且在千万级数据量的数据集上，速度提升明显。但算法在维度大于 5 的高维度数据聚类时算法效率存在不足，后期进一步研究解决。

参考文献：

[1] 唐东明，朱清新，杨凡，等. 一种有效的蛋白质序列聚类分析方法[J]. 软件学报，2011，22 (8)：1827-1837. (Tang Dongming, Zhu Qingxin, Yang Fan, *et al.* Efficient cluster analysis method for protein sequences [J]. Journal of Sofware, 2011, 22 (8): 1827-1837.)

[2] 苏阳阳，孙冬璞，李丹丹，等. 基于聚类和流量传播图的 P2P 流量识别方法 [J//OL]. 计算机应用研究, 2019 (11): 1-2 [2018-12-18]. (Su Yangyang, Sun Dongpu, Li Dandan, *et al.* P2P traffic identification method based on clustering and traffic dispersion graph [J/OL]. Application Research of Computers, 2019 (11): 1-2 [2018-12-18].)

[3] 邹永贵，万建斌，夏英. 基于路网的 LBSN 用户移动轨迹聚类挖掘方法 [J]. 计算机应用研究, 2013, 30 (8) : 2410-2414. (Zou Yonggui, Wan Jianbin, Xia Ying. LBSN user movement trajectory clustering mining method based on road network [J]. Application Research of Computers, 2013, 30 (8) : 2410-2414.)

[4] 吴明明，陈勇，房昊. 基于 K-均值聚类的彩色图像质量评价及优化 [J//OL]. 计算机应用研究，2019 (11): 1-6 [2018-12-18]. (Wu Mingming, Chen Yong, Fang Hao. Color image quality assessment and optimization based on K-mean clustering [J/OL]. Application Research of Computers, 2019 (11): 1-6 [2018-12-18].)

[5] MacQueen J. Some methods for classification and analysis of multivariate observations [C]// Proc of the 5th Berkeley Symposium on Mathematical Statistics and Probability. 1967: 281-297.

[6] 薛文娟，刘培玉，刘栋. 引入共享近邻加权图的 Chameleon 算法 [J]. 计算机应用, 2012, 32 (10): 2884-2887. (Xue Wenjuan, Liu Peiyu , Liu Dong. Improved Chameleon algorithm using weighted nearest neighbors graph [J]. Journal of Computer Applications, 2012, 32 (10): 2884-2887.)

[7] Ester M, Kriegel H P, Sander J, *et al.* A density-based algorithm for discovering clusters in large spatial databases with noise [C]// Proc of KDD. 1996: 226-231.

[8] Ankerst M, Breunig M M, Kriegel H P, *et al.* OPTICS: ordering points to identify the clustering structure [J]. ACM Sigmod Record, 1999, 28 (2): 49-60.

[9] Wei Wang , Yang Jiong, Muntz R. STING: a statistical information grid approach to spatial data mining [C]// Proc of VLDB. 1997: 186-195.

[10] Zhao Qinpei, Shi Yang, Liu Qin, *et al.* A grid-growing clustering algorithm for geo-spatial data [J]. Pattern Recognition Letters, 2015, 53 (53): 77-84.

[11] Yang Renyong, Xiong Tengke, Chen Tao, *et al.* DISTRIM: parallel GMM learning on multicore cluster [C]// Proc of IEEE International Conference on Computer Science and Automation Engineering . 2012, 2: 630-635.

[12] 于彦伟，贾召飞，曹磊，等. 面向位置大数据的快速密度聚类算法 [J]. 软件学报, 2018, 29 (8): 2470-2484. (Yu Yanwei, Jia Zhaofei, Cao Lei, *et al.* An efficient density-based clustering algorithm for location big data [J]. Journal of Software, 2018, 29 (8): 2470-2484.)

[13] Fu Limin, Enzo M. FLAME: a novel fuzzy clustering method for the analysis of DNA microarray data [J]. Bmc Bioinformatics, 2007, 8 (1): 3-0.

[14] Chang Hong, Yeung D. Robust path-based spectral clustering [J]. Pattern Recognition, 2008, 41 (1): 191-203.

[15] Zahn C T. Graph-theoretical methods for detecting and describing gestalt clusters [J]. IEEE Trans on Computers, 1970, 20 (SLAC-PUB-0672-REV): 68-86.

[16] Veenman C J, Reinders M J T, Backer E. A maximum variance cluster algorithm [J]. IEEE Trans on Pattern Analysis & Machine Intelligence, 2002, 24 (9): 1273-1280.

[17] Gionis A, Mannila H, Tsaparas P. Clustering aggregation [J]. ACM Trans on Knowledge Discovery from Data, 2007, 1 (1): 4.

[18] Karypis G, Eui-Hong H, Kumar V. Chameleon: hierarchical clustering using dynamic modeling [J]. Computer, 2002, 32 (8): 68-75.

[19] Zheng Yu, Zhang Lizhu, Xie Xing, *et al.* Mining interesting locations and travel sequences from GPS trajectories [C]//Proc of International Conference on World Wide Web. 2009: 791-800.

[20] Zheng Yu, Li Quannan, Chen Yunku, *et al.* Understanding mobility based on GPS data [C]// Proc of the 10th ACM International Conference on Ubiquitous Computing. 2008: 312-321.

[21] Zheng Yu, Xie Xing, Ma Wei-Ying. Geolife: a collaborative social networking service among user, location and trajectory [J]. IEEE Data(base) Engineering Bulletin, 2010, 33 (2): 32-39.

[22] Lyon R J, Stappers B W, Cooper S, *et al.* Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach [J]. Monthly Notices of the Royal Astronomical Society, 2016, 459 (1): 1104-1123.

附录：

附录 1 引理 1 证明

证明 设数据点 p 处于网格 G_l 中，其 Ssl 邻域内的所有数据点都为其邻居点，如图 1 所示，分别为 $Layer=3$、4、5 时的 $Grid$ 网格划分，在极端情况下 p 位于网格 G_l 的顶点上，但无论其处于哪个顶点（如蓝色点 p 和红色点 p 分别处于左上角和右下角），在 G_l 网格附近 $Layer-2$ 层内的网格（如图 1 中黑色框线所包括的网格）中的数据点都在其 Ssl 邻域内，即都为其邻居点。所以若 G_l 满足式（8），则点 p 一定是核心点。

附录 2 引理 2 证明

证明 数据点 p 在网格 G_l 中如果不是处于中心，则必然靠近于网格中的一个顶点，确定其所靠近的顶点后，则其邻域所能覆盖到的网格为其所在网格中心点 Ssl 邻域内的网格加上其靠近的顶点向外拓展的一层长度为 $2Layer$ 的范围内的网格，在二维中的表现如图 1 蓝点 p 和红点 p 的 Ssl 邻域所覆盖的网格所示，即至多统计该范围内的 $(2Layer)^k$ 个网格中为 p 的邻居点的数据点个数，因为邻域为方形的缘故，这一步并不需要计算距离，只需对边界网格内的各个数据点判断其各维数据是否在数据点 p 的网格区间内，就可以快速的判断范围内 p 的邻居点的个数。其中中心的 $(2Layer-3)^k$ 个网格只需要统计个数，无须判断，所以至多需要判断 $(2Layer)^k-(2Layer-3)^k$ 个网格。